REMARKS

By this Amendment, Applicant amends claims 3-5 and 13-16 and adds new claims 17-20. Claims 1-20 are therefore pending in this application.

In the final Office Action of January 30, 2006 ("Office Action"),[1] claims 1-4 were rejected under 35 U.S.C. § 112, second paragraph, as being indefinite; claims 1-16 were rejected under 35 U.S.C. § 102(a) as anticipated by the document entitled "64-Bit Application Development for PA-RISC & IA-64" ("*Coutant*"); and claims 1-4 and 13-16 were rejected under 35 U.S.C. § 102(b) as anticipated by the document entitled "*Microsoft* Interface Definition Language (MIDL): 64-Bit Porting Guide" ("*Microsoft*"). Applicant addresses the rejections and new claims below.

Section 112 rejection of claims 1-4

The Examiner alleged that claims 1-4 are indefinite because they include elements that have "insufficient antecedent [basis] . . . in the specification." Office Action at 4. In particular, the Examiner stated that the "adding . . . a directionality" and "adding . . . a parameter size" features of claim 1 were not found in the specification and therefore render claims 1-4 indefinite. Applicant traverses the § 112 rejection of claims 1-4 for at least the following reasons.

First, claims 3 and 4 do not depend upon claim 1 and do not recite the above-noted adding features of claim 1. As such, the § 112 rejection of claims 3 and 4 as set forth in the Office Action appears improper.

[1] The final Office Action contains a number of statements reflecting characterizations of the related art and the claims. Regardless of whether or not any such statement is identified herein, Applicant declines to automatically subscribe to any statement or characterization in the final Office Action. Furthermore, contrary to the Examiner's allegation, Applicant submits that the Amendment filed October 28, 2005, is fully compliant with 37 C.F.R. § 1.111(c) and M.P.E.P. § 714.04. Office Action at p. 4.

Second, the Examiner appears to be of the position that claims 1-4 are based on an insufficient disclosure and/or define an invention not described in the application as filed. To the extent the Examiner is in fact taking such a position, a rejection of the claims under § 112, second paragraph, is not the appropriate basis for rejection. *See* M.P.E.P. § 2174. Applicant notes that § 112, first paragraph, addresses the written description requirement. For this additional reason, the § 112 rejection is improper.

Third, Applicant's specification does in fact support the "adding" features noted in the Office Action. For example, Fig. 3A (element 334) depicts an "Insert Directionality" function. Additionally, the specification, on page 10, explains that "the interface generator inserts the directionality or intent of the parameter into the interface file . . . ." The specification also explains, on page 11, that "the interface generator inserts the size of the parameter along each of its dimensions . . . ." By virtue of at least these disclosures, the specification supports the "adding" features of claim 1.

Applicant submits that claims 1-4 are fully compliant with the requirements of § 112. Applicant therefore requests withdrawal of the § 112 rejection of these claims. In referring to the specification above, Applicant does not intend to limit the scope of the claims to the exemplary embodiments shown in the drawings and described in the specification. Rather, Applicant expressly affirms the entitlement to have the claims interpreted broadly, to the maximum extent permitted by statute, regulation, and applicable case law.

Section 102(a) rejection of claims 1-16

I. Regarding the *Coutant* document

The Examiner has not established that *Coutant* qualifies as prior art under 35 U.S.C. § 102(a). The *Coutant* document appears to be a printout of an electronic presentation and bears the date of March 17, 2000. To the extent the Examiner is applying *Coutant* as a "printed publication" within the meaning of § 102(a), Applicant again calls attention to the requirement for "a satisfactory showing that such document has been disseminated or otherwise made available to the extent that persons interested and ordinarily skilled in the subject matter or art, exercising reasonable diligence, can locate it." M.P.E.P. § 2128 (internal citations omitted). Further, an electronic publication cannot be relied upon as prior art under § 102(a) if it does not include a publication date or retrieval date. *See* M.P.E.P. § 2128. In this case, the Examiner has not established that the date of March 17, 2000, is relevant to the *Coutant*'s availability, publication, or retrieval.

In the Office Action, the Examiner alleged that "Hewlett Packard and Microsoft had circulated and discussed the 'conversion of 32-bit and 64-bit' long . . . before [Applicant's] . . . filing date" and that "the submitted prior [art] . . . are typical examples . . . ." Office Action at 2. According to the Examiner, *Coutant*'s "discussion [the source of which is Hewlett Packard] . . . dated on 3/17/2000 . . . [is] sufficient [proof]" of such circulation and discussion. *Id.* The Examiner also cited M.P.E.P. § 2128 and case law regarding the meaning of a "printed publication" within the context of § 102. *See id.* at 3. For example, the Examiner noted that "the key inquiry is whether or not a reference has been made publicly accessible." *Id.* (citing *In re Klopfenstein*, 380 F.3d 1345, 1348, 72 USPQ2d 1117, 1119 (Fed. Cir. 2004)). The Examiner also noted various factors used to determine whether "a temporarily displayed reference that was neither distributed nor indexed" constitutes a "printed publication" within the context of § 102. *Id.* (citing *In re Klopfenstein*, 380 F.3d at 1350). The Examiner, however, offered no evidence or explanation showing that *Coutant* was in fact publicly accessible. The bare allegation that Hewlett Packard and Microsoft and their supposed "online libraries" show "steady efforts for developments of 32 bit/64 bit conversions before the filing of this application" is not a sufficient showing of *Coutant*'s dissemination and accessibility. *Id.* at 2. Moreover, the Examiner provided no evidence or explanation as to the above-noted factors cited in the Office Action. Unless the Examiner produces the requisite proof of *Coutant*'s dissemination and availability prior to Applicant's filing date (*See* M.P.E.P. § 2128), *Coutant* is not a competent prior art reference within the context of § 102(a) and cannot be used to reject Applicant's claims.

II. Regarding the merits of the rejection

Regardless of whether *Coutant* is in fact a proper § 102(a) reference, Applicant traverses the § 102(a) rejection of claims 1-16 because *Coutant* fails to anticipate these claims. In order to properly anticipate Applicant's claimed invention under § 102, each and every element of the claim at issue must be found, either expressly described or under principles of inherency, in a single prior art reference. Further, "[t]he identical invention must be shown in as complete detail as is contained in the . . . claim." *See* M.P.E.P. § 2131. Finally, "[t]he elements must be arranged as required by the claim." *Id.*

With regard to claim 1, *Coutant* fails to teach at least the following features:

> adding to the interface file a directionality of at least one of the integer parameter and the logical parameter based on comments in the source code;
>
> adding to the interface file a parameter size along each dimension of at least one of the integer parameter and the logical parameter; and
>
> reading the interface file to generate a stub routine that converts at least one of the integer and logical parameters from 32-bit to 64-bit and that invokes the subprogram by specifying the converted parameters.

Coutant describes a 64-bit programming model. It addresses porting to 64-bits, as well preparing code for 64 bits. *Coutant* compares ILP32 (integer, long, and pointer) and the LP64 programming models, noting that in LP64 long and pointer types are 64 bits and also that LP64 includes certain extended derived types (pages 3-4). *Coutant* also compares aspects of 32-bit and 64-bit runtime environments, commenting that with 64-bit code the compiler can inline import stubs (pages 5-6). *Coutant* does not support the § 102(a) rejection of claim 1, as discussed below.

In alleging that *Coutant* teaches the above-noted "adding" features of claim 1, the Examiner notes *Coutant*'s disclosure regarding the various base types (e.g., *int* and *long*) and their respective bit size (e.g., 32-bit or 64-bit) in the ILP32 and LP64 programming models. Office Action at 5 (citing *Coutant* at 3-4). The Examiner also noted that *Coutant* mentions checking the usage of the *int* and *long* types for consistency (page 14) and accessing a "linkage table" in a program for globals (page 11). Office Action at 5. The mere comparison of *int*, *long*, and other base types in different programming models does not teach "adding to the interface file a directionality of at least one of the integer parameter and the logical parameter based on comments in the source code," as recited in claim 1. Even if indicating an *int*, *long*, or other type were construed as indicating a "directionality" of a parameter in source code, Applicant disputing such a construction, *Coutant* does not disclose that such an indication would be added to an "interface file . . . based on comments in the source code," as claimed.

Coutant's comparison of types further fails to teach "adding to the interface file a parameter size along each dimension of at least one of the integer parameter and the logical parameter," as recited in claim 1. The "sizes" identified by *Coutant* reflect the bit length of base types in the in the ILP32 and LP64 models. *Coutant* does not teach a "parameter size along each dimension" of a parameter. Further, comparing respective bit lengths for types in the ILP32 and LP64 models does not constitute "adding to the interface file a parameter size along each dimension," as claimed. Even if respective bit length for a type in the ILP32 and LP64 programming models were construed as a "parameter size," Applicant disputing such a construction, *Coutant* does not disclose that an indication of the bit length would be added to an "interface file."

Furthermore, the mere idea that the usage of the *int* and *long* types might be checked for consistency does not teach either of the claimed "adding" features. Additionally, the mere usage of a "linkage table" fails to support the Examiner's position that *Coutant* teaches the claimed "adding" features. Indeed, neither the relied-upon portions nor any other portions of *Coutant* teach the "adding" features of claim 1.

In alleging that *Coutant* teaches the claimed "reading" feature of claim 1, the Examiner notes *Coutant*'s disclosure that, in 64-bit code, the compiler can inline import stubs (page 5). Office Action at 5. The Examiner also notes *Coutant*'s disclosure regarding eliminating "parameter relocation" (page 6). *Id.* Neither these portions nor any other portions of *Coutant*, however, teach "reading the interface file to generate a stub routine that converts at least one of the integer and logical parameters from 32-bit to 64-bit and that invokes the subprogram by specifying the converted parameters," as recited in claim 1. Although *Coutant* mentions "import stubs" and eliminating "parameter relocation," the reference does not teach "reading an interface file," as claimed.

Applicant reminds the Examiner that a rejection under § 102 is proper only when the claimed subject matter is <u>identically</u> described or disclosed in the prior art. *In re Arkley*, 455 F.2d 586, 587, 172 USPQ 524, 526 (CCPA 1972). As noted above, "[t]he identical invention must be shown in as complete detail as is contained in the . . . claim." M.P.E.P. § 2131. In this case, *Coutant* does not identically describe the subject matter of claim 1 "in as complete detail as is contained in the . . . claim."

Furthermore, even if all of the features of claim 1 could be found in various teachings of *Coutant* – Applicant disputing that contention – the reference does not clearly and unequivocally disclose the claimed invention or direct those skilled in the art to the claimed invention "<u>without any need for picking, choosing, and combining</u> various disclosures." *In re Arkley*, 455 F.2d 586, 587,172 USPQ 524, 526 (CCPA 1972) (emphasis added).

For at least the foregoing reasons, *Coutant* does not support the § 102(a) rejection of claim 1, as asserted by the Examiner. As such, the § 102(a) rejection of claim 1 based on *Coutant* should be withdrawn.

With regard to independent claim 3, *Coutant* fails to teach at least "generating, from the 32-bit source code, a 32-bit interface file including statements describing characteristics of parameters in the 32-bit source code" and "automatically generating, based on the statements in the 32-bit interface file, a 32-bit to 64-bit conversion stub that is used by the 32-bit source code to invoke 64-bit code," as recited in claim 3. Although *Coutant* compares 32-bit and 64-bit programming models, the reference does not teach at least "generating . . . a 32-bit interface file" and "automatically generating a 32-bit to 64-bit conversion stub," as claimed. *Coutant*'s description of import stubs and eliminating "parameter relocation" (pages 5-6) does not teach "generating . . . a 32-bit interface file" and "automatically generating a 32-bit to 64-bit conversion stub," as claimed. Because *Coutant* does not teach each and every feature of claim 3, as a matter of law, it cannot anticipate this claim. As such, the § 102(a) rejection of claim 3 based on *Coutant* should be withdrawn.

With regard to independent claim 5, *Coutant* fails to teach at least the following features:

> an interface generator that reads the subprogram and that generates an interface file with indications of characteristics of the parameter; and
>
> a stub generator that reads the interface file and that generates a stub for the subprogram by using the characteristics, wherein each of the stubs receives a set of parameter values, generates the values for the required parameters from the received set of parameter values, and invokes the subprogram with the values for the parameters.

In rejecting claim 5, the Examiner again noted *Coutant*'s comparison of the ILP32 and LP64 programming models. Office Action at 6 (citing *Coutant* at 3-4). The Examiner alleged that in *Coutant* "the 'type' is identified as characteristics for the developing of 32-bit code and 64-bit code in conversing/porting." *Id.* Applicant disagrees with the Examiner's interpretation of *Coutant*. To begin with, even if *Coutant* were to disclose parameter "characteristics," the document does not teach "an interface generator that reads the subprogram and that generates an interface file with indications of characteristics of the parameter" and "a stub generator that reads the interface file and that generates a stub for the subprogram by using the characteristics . . .," as asserted by the Examiner. Instead, *Coutant* merely compares ILP32 and LP64, noting that in LP64 long and pointer types are 64 bits and also that LP64 includes certain extended derived types (pages 3-4). Further, *Coutant* does not disclose "developing" 32-bit code or converting between 64-bit and 32-bit code, and the Examiner provides no evidence to support the allegation that the identified "types" facilitate such developing and converting. Additionally, *Coutant*'s comparison of the 32-bit and 64-bit runtime environments at pages 5 and 6 does not constitute "an interface generator" and "a stub generator," as recited in claim 5. For at least the foregoing reasons, *Coutant* does not support the § 102(a) rejection of claim 5. The § 102(a) rejection of claim 5 based on *Coutant* should therefore be withdrawn.

Independent claims 13 and 16, although different in scope from claim 3 (and from each other), include features similar to those of claim 3. In particular, claim 13 recites, *inter alia*, "generating, from the 32-bit source code, a 32-bit interface file including statements describing characteristics of parameters in the 32-bit source code" and "automatically generating a 32-bit interface to 64-bit source code based on the statements in the interface file." Claim 16 recites, *inter alia*, "means for generating, from the 32-bit source code, a 32-bit interface file including statements describing characteristics of parameters in the 32-bit source code" and "means for automatically generating, based on the statements in the 32-bit interface file, a 32-bit to 64-bit conversion stub that is used by the 32-bit source code to invoke 64-bit code." For at least reasons similar to those presented above in connection with claim 3, claims 13 and 16 are distinguishable from *Coutant*. Accordingly, the § 102(a) rejection of claims 13 and 16 should be withdrawn.

As to independent claim 15, *Coutant* fails to teach at least "generating from the source code an interface file including characteristics of the parameter" and "generating, based on the characteristics of the parameter, a stub routine that invokes the subprogram and that facilitates use of at least one of a converted integer and logical parameter," as claimed. *Coutant*'s comparison of the 32-bit and 64-bit runtime environments at pages 5 and 6 does not teach the above-noted "generating" features of claim 15. Indeed, neither the cited portions nor any other portions of *Coutant* teach the above-noted "generating" features. Because *Coutant* does not teach each and every feature of claim 15, as a matter of law, it cannot anticipate this claim. As such, the § 102(a) rejection of claim 15 based on *Coutant* should be withdrawn.

Claim 2 depends upon claim 1, claim 4 depends upon claim 3, claims 6-12 depend upon claim 5, and claim 14 depends upon claim 13. Claims 2, 4, 6-12, and 14 are distinguishable from

Coutant for at least reasons similar to those presented above in connection with claims 1, 3, 5, and 13. The § 102(a) rejection of dependent claims 2, 4, 6-12, and 14 should therefore be withdrawn.

In addition, with respect to dependent claim 4, *Coutant* fails to teach at least "determining whether the . . . parameter has input directionality, output directionality, or input and output directionality" and "inserting into the 32-bit interface file code generator statements corresponding to the determined directionality," as claimed. *Coutant* also fails to teach at least "determining whether the parameter in the subprogram has input directionality, output directionality, or input and output directionality" and "inserting into the 32-bit interface file code generator statements corresponding to the determined directionality," as recited in dependent claim 14. For these additional reasons, claims 4 and 14 are distinguishable from *Coutant* and the § 102(a) rejection of the claims should be withdrawn.

Section 102(b) rejection of claims 1-4 and 13-16

I. Regarding the *Microsoft* document

The Examiner has not established that *Microsoft* qualifies as prior art under 35 U.S.C. § 102(b). The *Microsoft* document, which appears to be a printout of an Internet file presumably extracted from a larger work, bears a "Last updated" date of August 1999 and refers to a technical white paper, which appears to claim a copyright date of 1998 (*see* page 18). To the extent the Examiner is applying *Microsoft* as a "printed publication," Applicant again calls attention to the requirement for "a satisfactory showing that such document has been disseminated or otherwise made available to the extent that persons interested and ordinarily skilled in the subject matter or art, exercising reasonable diligence, can locate it." M.P.E.P. § 2128 (internal citations omitted). Further, an electronic publication cannot be relied upon as prior art under if it does not include a publication date or retrieval date. *See* M.P.E.P. § 2128. The Examiner has not established *Microsoft*'s 1998 and 1999 dates are relevant to its availability, publication, or retrieval, as discussed below.

As noted above in connection with the *Coutant* reference, the Examiner alleged that "Hewlett Packard and Microsoft had circulated and discussed the 'conversion of 32-bit and 64-bit' long . . . before [Applicant's] . . . filing date" and that "the submitted prior [art] . . . are typical examples . . . ." Office Action at 2. According to the Examiner, the "Last updated" date of August 1999 appearing on the *Microsoft* document is "sufficient" proof of such circulation and discussion. *Id.* As noted above, the Examiner also cited the M.P.E.P. and case law regarding the meaning of a "printed publication" within the context of § 102. *See id.* at 3. The Examiner, however, offered no evidence or explanation showing that *Microsoft* was in fact publicly accessible. The bare allegation that Hewlett Packard and Microsoft and their supposed "online libraries" show "steady efforts for developments of 32 bit/64 bit conversions before the filing of this application" is not a sufficient showing of the *Microsoft* document's dissemination and accessibility. *Id.* at 2. Moreover, the Examiner provided no evidence or explanation as to the various factors cited in the Office Action. *Id.* at 3 (citing *In re Klopfenstein*, 380 F.3d at 1350). Unless the Examiner produces the requisite proof of *Microsoft*'s dissemination and availability more than one year prior to Applicant's filing date (*See* M.P.E.P. § 2128), *Microsoft* is not a competent prior art reference within the context of § 102(b) and cannot be used to reject Applicant's claims.

II. Regarding the merits of the rejection

Microsoft's status as a proper § 102(b) reference aside, Applicant traverses the § 102(b) rejection of claims 1-4 and 13-16 because *Microsoft* fails to anticipate these claims.

With regard to claim 1, *Microsoft* fails to teach at least the "adding" features. The *Microsoft* reference is an MIDL 64-bit porting guide. The guide describes "new 64-bit features that can facilitate porting applications from 32-bit environments" (page 1). In rejecting claim 1, the Examiner alleged that *Microsoft* discloses "using MIDL as a programming language for [converting] 32-bit to 64-bit." Office Action at 9. The Examiner also noted *Microsoft*'s disclosure regarding "the syntax of subroutine calls and Handle Types of MIDL" and alleged that *Microsoft* discloses "an IDL setup . . . [that] shows adding parameter lists." *Id.* (citing *Microsoft* at 12). *Microsoft* does not support the § 102(b) rejection of claim 1, as discussed below.

To begin with, the Examiner provided no evidence showing that *Microsoft* teaches "using MIDL as a programming language for [converting] 32-bit to 64-bit." Indeed, *Microsoft* merely addresses "features that can facilitate porting applications from 32-bit environments (page 1)." Furthermore, contrary to the Examiner's position, *Microsoft*'s disclosure regarding a "typical IDL setup" does not constitute "adding to the interface file a directionality of at least one of the integer parameter and the logical parameter based on comments in the source code" or "adding to the interface file a parameter size along each dimension of at least one of the integer parameter and the logical parameter," as recited in claim 1. *Microsoft*'s IDL setup on page 12 merely references a <users parameter list> and a <remotable parameter list>. Even if *Microsoft*'s IDL specification were construed as an "interface file," to which Applicant does not acquiesce, *Microsoft* does not teach adding to the IDL specification a "directionality" of a parameter, let alone adding such directionality "based on comments in the source code," as claimed. Likewise, *Microsoft* does not teach adding to the IDL specification "a parameter size along each dimension" of a parameter, as claimed. Indeed, *Microsoft* fails to teach the "adding" features of claim 1.

Microsoft further fails to teach at least the "reading" feature of claim 1. The Examiner alleged that "when a compiler [reads] . . . the interface definition written under MIDL, it calls a 64-bit stub generation, for example, . . . the 64b type libraries are supported by mapping 32b *.TLB files." Office Action at 9 (citing *Microsoft* at 15). That 64b type libraries might be supported by mapping 32b type library files (TLBs) does not establish that a compiler "calls a 64-bit stub generation," as alleged by the Examiner. Further, such mapping does not teach "reading the interface file to generate a stub routine that converts at least one of the integer and logical parameters from 32-bit to 64-bit and that invokes the subprogram by specifying the converted parameters," as recited in claim 1. Even if a compiler were to call "a 64-bit stub generation," as alleged by the Examiner, such functionality does not constitute "reading the interface file to generate a stub routine that converts at least one of the integer and logical parameters from 32-bit to 64-bit and that invokes the subprogram by specifying the converted parameters," as claimed. Contrary to the Examiner's position, merely calling a stub generation does not teach generating a stub routing that converts parameters and invokes a subprogram by specifying the converted parameters. *See* Office Action at 9. Indeed, *Microsoft* fails to teach the "reading" feature of claim 1.

For at least the foregoing reasons, *Microsoft* does not support the § 102(b) rejection of claim 1, as asserted by the Examiner. As such, the § 102(b) rejection of claim 1 based on *Microsoft* should be withdrawn. The § 102(b) rejection of claim 2 based on *Microsoft* should be withdrawn as well, since claim 2 depends upon claim 1 and is likewise distinguishable from the applied reference.

As to independent claim 3, *Microsoft* fails to teach at least "generating . . . a 32-bit interface file" and "automatically generating . . . a 32-bit to 64-bit conversion stub," as claimed.

Although *Microsoft* describes a 64-bit stub generation model and porting applications from 32-bit environments (e.g., pp. 1, 15), the reference fails to teach "generating, from the 32-bit source code, a 32-bit interface file including statements describing characteristics of parameters in the 32-bit source code" and "automatically generating, based on the statements in the 32-bit interface file, a 32-bit to 64-bit conversion stub that is used by the 32-bit source code to invoke 64-bit code," as claimed. *Microsoft* describes a compiler generating a stub for a given environment (page 15). *Microsoft* also describes a "dual stub," which consists of 32-bit and 64-bit parts, for use in 32-bit and 64-bit environments (page 15). *Microsoft* mentions that dual stub files are "a convenient tool for porting . . ." (page 15). Generating a dual stub for use in a 32-bit environment and a 64-bit environment, which may facilitate porting, does not constitute "generating, from the 32-bit source code, a 32-bit interface file including statements describing characteristics of parameters in the 32-bit source code" and "automatically generating, based on the statements in the 32-bit interface file, a 32-bit to 64-bit conversion stub that is used by the 32-bit source code to invoke 64-bit code," as claimed. Even if *Microsoft*'s "dual stub" were construed as a "32-bit to 64-bit conversion stub," Applicant disputing such a construction, *Microsoft* does not disclose generating the dual stub automatically based on "statements [describing characteristics of parameters in 32-bit source code] in . . . [a] 32-bit interface file" generated from the 32-bit source code, as claimed. Indeed, *Microsoft* fails to teach at least the "generating . . . a 32-bit interface file" and "automatically generating . . . a 32-bit to 64-bit conversion stub" features of claim 3.

Because *Microsoft* does not teach each and every feature of claim 3, as a matter of law, it cannot anticipate this claim. The § 102(b) rejection of claim 3 based on *Microsoft* should therefore be withdrawn.

Claim 4 depends upon claim 1 and is likewise distinguishable from *Microsoft*. *Microsoft* further fails to teach at least "determining whether the . . . parameter has input directionality, output directionality, or input and output directionality" and "inserting into the 32-bit interface file code generator statements corresponding to the determined directionality," as recited in dependent claim 4. The § 102(b) rejection of claim 4 based on *Microsoft* should therefore be withdrawn.

Independent claim 13, although of different scope than claim 3, includes features similar to those of claim 3. In particular, claim 13 recites, *inter alia*, "generating, from the 32-bit source code, a 32-bit interface file including statements describing characteristics of parameters in the 32-bit source code" and "automatically generating a 32-bit interface to 64-bit source code based on the statements in the interface file." For at least reasons similar to those presented above in connection with claim 3, claim 13 is distinguishable from *Microsoft*. Although *Microsoft* discloses (page 1) "64-bit features that can facilitate porting applications from 32-bit environments," the reference does not teach at least the above-noted features of claim 13. Accordingly, the § 102(b) rejection of claim 13 based on *Microsoft* should be withdrawn.

The Examiner alleged that "the functionality of each of independent . . . [claims] 14-16 corresponds to functionality of claim 13" and rejected claims 14-16 for the same reasons set forth in the rejection of claim 13. Office Action at 10. The § 102(b) rejection of claims 14-16 should be withdrawn for at least the following reasons.

Initially, Applicant notes that claim 14 is <u>not</u> an independent claim and includes additional features not recited in claim 13. For at least this reason, the § 102(b) rejection of claim 14 is improper. Moreover, claim 14 depends upon claim 13 and is likewise distinguishable from *Microsoft*. *Microsoft* further fails to teach at least "determining whether the parameter in the subprogram has input directionality, output directionality, or input and output directionality" and "inserting into the 32-bit interface file code generator statements corresponding to the determined directionality," as recited in dependent claim 14. Accordingly, the § 102(b) rejection of claim 14 based on *Microsoft* should be withdrawn. Should the Examiner continue to dispute the patentability claim 14, Applicant requests clarification in the next Action (which should be non-final) as to the basis for such a position.

With regard to independent claim 15, *Microsoft* fails to teach at least "generating from the source code an interface file including characteristics of the parameter" and "generating, based on the characteristics of the parameter, a stub routine that invokes the subprogram and that facilitates use of at least one of a converted integer and logical parameter," as claimed. Although *Microsoft* describes a 64-bit stub generation model and porting applications from 32-bit environments, the reference does not teach at least the above-noted features of claim 15. *Microsoft*'s disclosure regarding "dual stubs" likewise fails to teach the above-noted "generating features" of claim 15. The dual stubs are not generated based on characteristics of a parameter in an interface file generated from the source code. Because *Microsoft* does not teach each and every feature of claim 15, as a matter of law, it cannot anticipate this claim. The § 102(b) rejection of claim 15 based on *Microsoft* should therefore be withdrawn.

Independent claim 16, although of different scope than claims 3 and 13, includes features similar to those of claim 3 and 13. For example, claim 16 recites, *inter alia*, "means for generating, from the 32-bit source code, a 32-bit interface file including statements describing characteristics of parameters in the 32-bit source code" and "means for automatically generating, based on the statements in the 32-bit interface file, a 32-bit to 64-bit conversion stub that is used by the 32-bit source code to invoke 64-bit code." For at least reasons similar to those presented above in connection with claims 3 and 13, claim 16 is distinguishable from *Microsoft*. Accordingly, the § 102(b) rejection of claim 16 based on *Microsoft* should be withdrawn.

New claims 17-20

New claims 17 and 18 depend upon claim 1, and new claims 19 and 20 depend upon claim 3. For at least reasons similar to those presented above in connection with claims 1 and 3, respectively, *Coutant* and *Microsoft* fail to teach or suggest each and every feature required by new claims 17-20. Additionally, the applied art fails to teach or suggest the additional features recited in new claims 17-20. Applicant thus requests the timely allowance of new claims 17-20.

Conclusion

Applicant requests the Examiner's reconsideration of the application in view of the foregoing, and the timely allowance of pending claims 1-20.

Please grant any extensions of time required to enter this response and charge any additional required fees to our deposit account 06-0916.

Respectfully submitted,

FINNEGAN, HENDERSON, FARABOW,
GARRETT & DUNNER, L.L.P.

Dated: May 1, 2006

By: [illegible]
Frank A. Italiano
Reg. No. 53,056